

Classes, Objetos, Abstração e Encapsulamento

Tópicos

1. Conceitos Gerais:

- Lei de Moore
- Dados;
- Informação;
- Sistema de Informação vs Software;
- Programação Estruturada e Paradigma da Orientação a Objeto.

2. Objeto, atributo e método.

3. Classe

4. Pilares da Orientação a Objeto:

- Abstração;
- Encapsulamento;
- Herança;
- Polimorfismo.

5. Exercícios

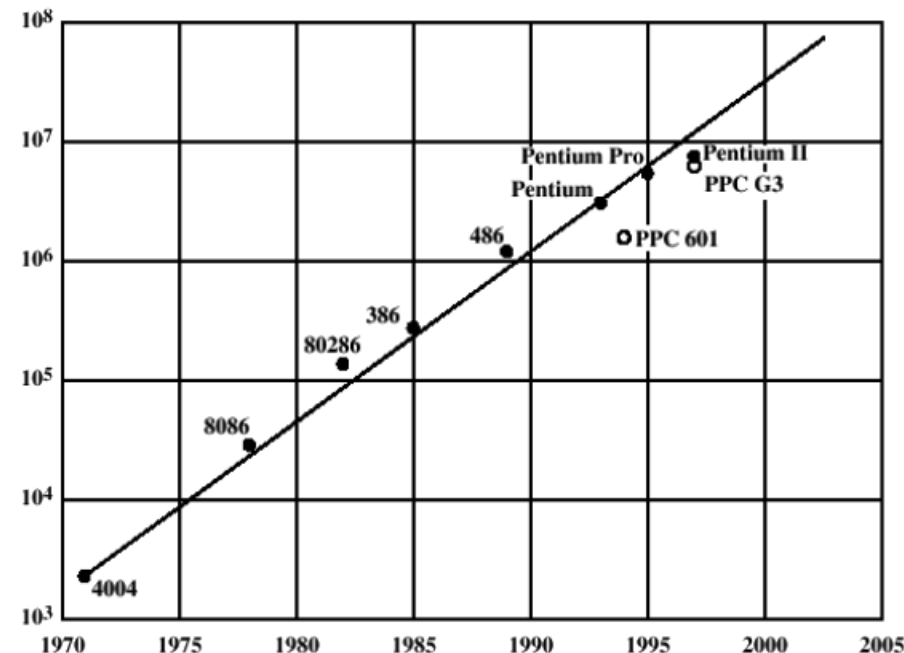
Conceitos gerais

Introdução

▪ LEI DE MOORE.

A Lei de Moore (1965), diz que o número de transistores em um chip de computador dobraria a cada ano, impulsionando o avanço tecnológico. Isso resultou em computadores mais poderosos e acessíveis ao longo do tempo.

Obs.: Profetizou que a quantidade de transistores que poderiam ser colocados em uma mesma área dobraria a cada 18 meses mantendo-se o mesmo custo de fabricação.



Introdução

DADOS: São fatos em uma forma primária, que podem ser armazenados em algum meio.

Exemplo

CPF, Nome, Data

INFORMAÇÃO: São os fatos organizados de maneira a produzir um significado. Dados colocados em contextos geram a informação.

Exemplo

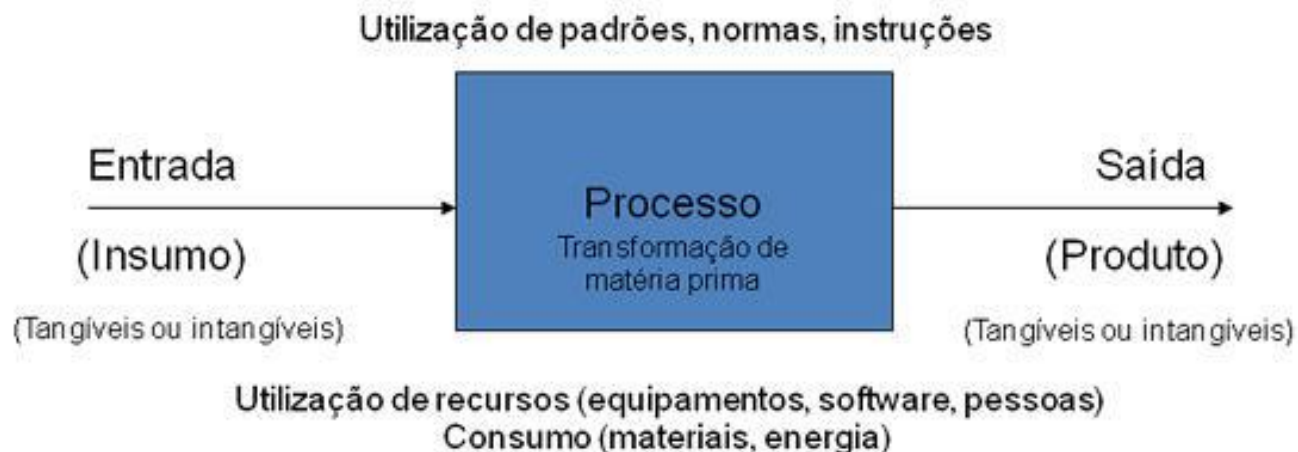
Lista de clientes com seus números de CPF, ordenados.

OBS.: Quando os dados são associados a outros e interpretados, eles dão origem a INFORMAÇÃO.

Processos de Negócios

Um processos é qualquer trabalho, operação administrativa, função biológica, produtiva, social, etc, pode ser considerado um processo.

Conjunto inter-relacionado de recursos e atividades seguindo um algoritmo pré-estabelecido que transformam entradas (insumos) em saídas (produtos).



Sistemas de Informação X Software

- **Sistemas de Informação:** Lidam com a gestão e o uso de informações dentro de uma organização para alcançar seus objetivos.
- **Software:** Refere-se a programas de computador, instruções ou dados que controlam o funcionamento de sistemas de computador e hardware.

Processo de desenvolvimento

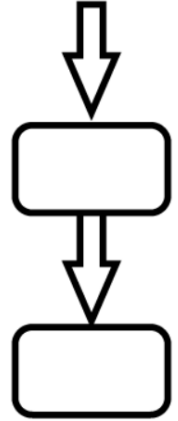
- Modelar um Sistema de Informação não é tarefa das mais simples e, dentro disso tudo, é importante pensarmos qual parte do processo seria automatizada por um sistema de software.
- Projetar um sistema de *software* compreende termos a noção de que ele é um componente de um Sistema de Informação.
- Diversas foram as abordagens e técnicas criadas para modelar sistemas de *software* até chegarmos à abordagem foco desta disciplina: Orientação a Objetos.

Programação Estruturada

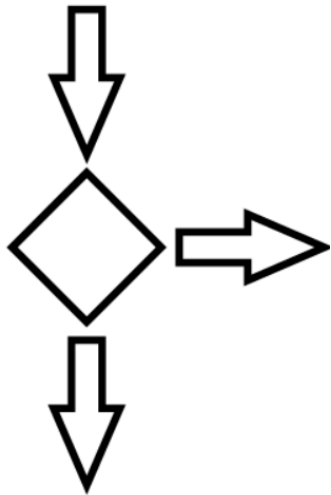
Na **programação estruturada**, é como tentar construir uma casa usando apenas tijolos soltos. Você tem os **dados (tijolos)** e as **funcionalidades (muros, janelas, portas)**, mas conectá-los de forma coesa pode ser um desafio.

Programação Estruturada

Sequência



Decisão



Estruturada - Este tipo de programação segue uma lógica e sequência de pensamentos de uma máquina. Ou seja, a sua lógica é direcionada à linguagem de máquina, já que ela realiza o que foi orientado pelo programador por meio de uma linguagem estruturada.

Programação OO

Na **orientação a objetos**, é como montar uma casa usando peças de LEGO. Cada **peça (objeto)** tem sua função específica e encaixa perfeitamente com outras peças, criando uma estrutura sólida e fácil de entender.



Diferença

- A POO possui algumas vantagens sobre a programação estruturada (PE):
 - Em PE, é difícil criar uma conexão forte entre dados e funcionalidades;
 - Manutenção, Produtividade, Reuso de código.

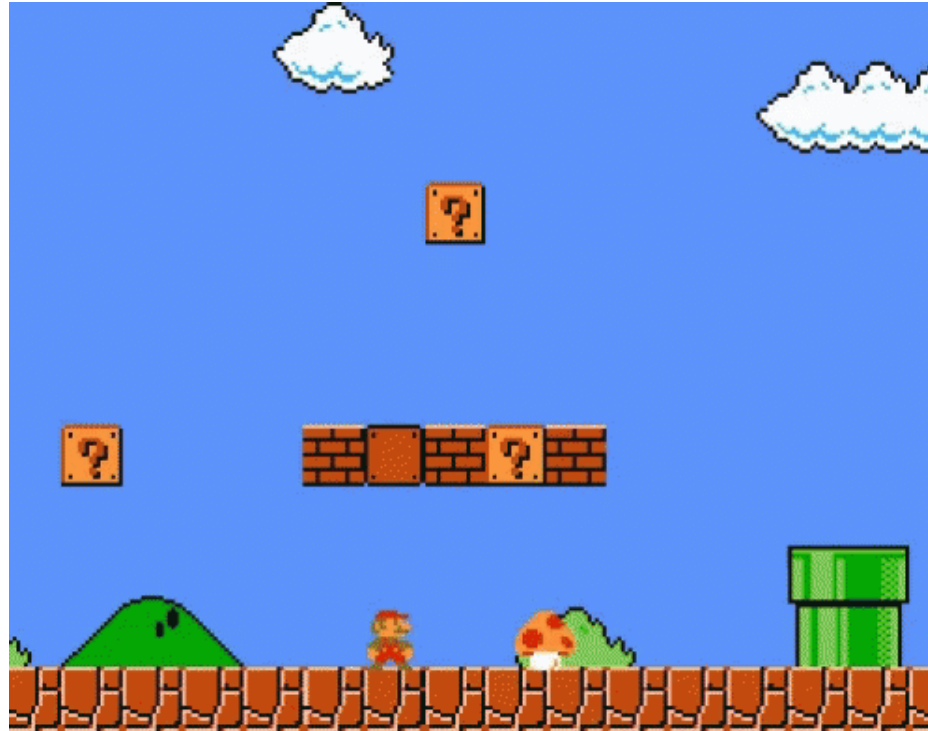
OBS.: Com a abordagem orientada a objetos, é como trocar uma peça de LEGO por outra - rápido, fácil e sem quebrar nada. Já na programação estruturada, pode ser mais parecido com tentar mudar um tijolo em um muro sem desmoroná-lo.

Objetos

Paradigma Orientado a Objetos

- Podemos relacionar esse paradigma de programação com o princípio de “imitar o mundo real”:
 - Nossa vida está repleta de **objetos físicos** (carro, livro, impressora, etc) e **abstratos** (pergunta, esquema tático, plano de aulas, etc);
 - Objeto é tudo aquilo que tem **atributos** e **comportamentos**.

Análise de Sistemas Orientado a Objetos (OO)



No jogo do Mario, **você pode pensar em cada elemento do jogo como um objeto**. temos o próprio Mario, os inimigos, os itens como moedas e cogumelos, os blocos que podem conter itens ou esconder passagens secretas etc.

Conceitos

you precisa compreender alguns conceitos! Vamos lá?

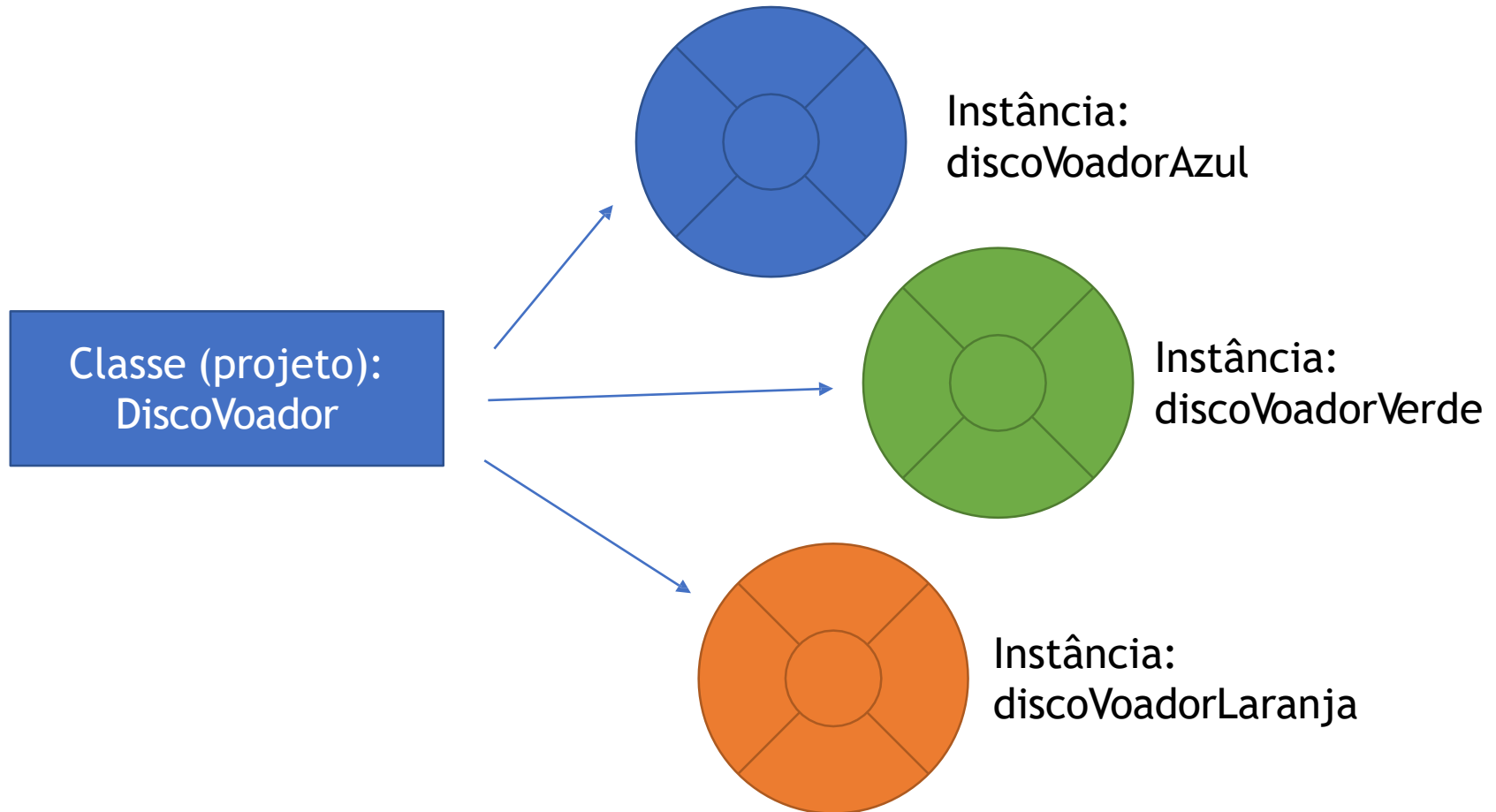
Objeto: São elementos do mundo real para o mundo da programação. O objeto é criado a partir de uma classe. É algo que se visualiza, se utiliza e assume um papel no domínio do problema. Um exemplo disso, é um Fusca 1964.

Atributos: São as características do objeto. Considerando o exemplo do Fusca 1964, os atributos deles são: ano de fabricação é 1964 e que ele tem um formato arredondado e o motor fica na parte traseira.

Métodos: São os comportamentos/ações do objeto. Ainda de acordo com o exemplo, o Fusca pode se locomover, transportando pessoas, cargas etc.

Vamos instanciar um disco voador?

- Um **objeto** é uma **instância** de uma **classe**.



Classes

Paradigma Orientado a Objetos

- Os programas são organizados por elementos chamados **classes**;

Imagine que estamos construindo uma fábrica de carros. Uma classe na OO é como o projeto ou o molde para fabricar esses carros.

Exemplo

Carro específico: Sedan (Esse sedan seria uma instância, ou seja, um objeto criado a partir do molde da classe de carros.)

Paradigma Orientado a Objetos

- Dentro da classe de carros, **temos todas as características comuns a todos os carros**, como o número de portas, o tipo de motor, a cor padrão, etc. Essas características são como variáveis dentro da classe.
- Também pode conter os **comportamentos** que todos os carros devem ter, como ligar o motor, acelerar, frear, etc. Esses comportamentos são como **métodos** dentro da classe.

Classe Livro

Classes:

| Livro | |
|------------|------------------------|
| Atributos: | Título: string |
| | Autor: string |
| | Ano de Publicação: int |
| | Editora: string |
| | Exemplares: int |
| Métodos: | Reservar() |
| | Emprestar() |
| | Devolver() |

Classe Empréstimo

Classes:

| Empréstimo | |
|------------|---------------------------------|
| Atributos: | Data de Empréstimo: date |
| | Data de Devolução: date |
| | Multa: float |
| | Usuário: reference (associação) |
| | Livro: reference (associação) |
| Métodos: | CalcularMulta() |

Resumo


Uma classe na orientação a objetos é como um modelo que define tanto as **características (variáveis)** quanto os **comportamentos (métodos)** que os objetos criados a partir dela terão.

Quando criamos objetos a partir de uma classe, estamos basicamente dando vida a esse modelo, construindo **instâncias (objetos)** que herdam todas as características e comportamentos definidos na classe.

Pilares

Paradigma da Orientação a Objetos (POO)

- Conceitos importantes em OO:

- Abstração
 - Encapsulamento
 - Herança
 - Polimorfismo
- 
- Pilares da OO

Abstração

Classe

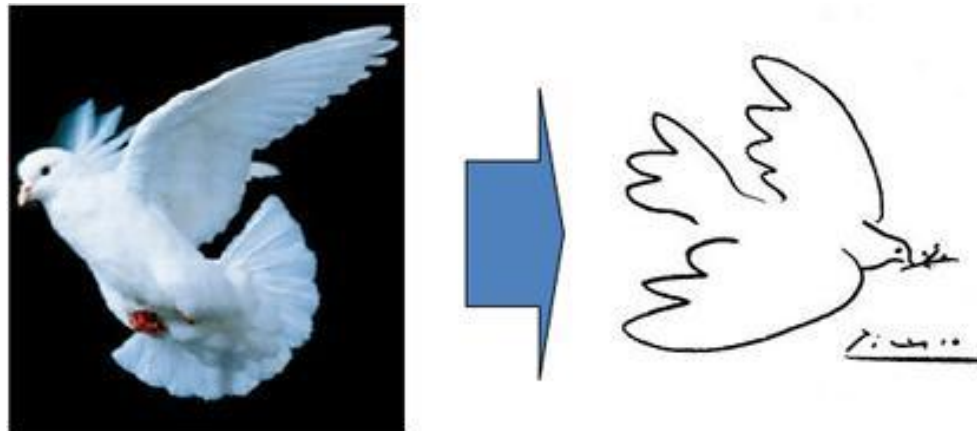
- **Abstração de um objeto:**
 - Damos o nome de abstração para o processo de aproximar o mundo real do mundo da programação, sendo o seu objetivo, simplificar um problema difícil. Para isso, a abstração leva em conta os aspectos importantes;
 - Na abstração, desconsideramos detalhes irrelevantes.

Exemplo

```
public class DiscoVoador {  
  
}
```


Classe

Abstração: Assim como representado na imagem, a abstração é a criação de uma classe abstrata, que é uma classe incompleta, como se fosse um quebra-cabeça. Assim, esta classe não permite a criação de instâncias e obriga a implementação de todos os métodos da classe assinados como 'abstract'.




Abstração

Ao criar uma classe para representar um objeto, como um carro ou um animal, você identifica as características essenciais desse objeto, como cor, tamanho, comportamentos etc.



Você não precisa se preocupar com todos os detalhes internos de implementação do objeto, como o funcionamento do motor do carro ou a biologia completa do animal.



Esses detalhes são abstraídos, ou seja, deixados de lado temporariamente, para se concentrar apenas nas características importantes para o contexto do problema que está sendo resolvido.

Na prática

Vamos supor que você está desenvolvendo um código com a classe “ser humano”. Essa classe é composta por inúmeras variantes (atributos), como por exemplo: altura, peso, cor da pele, cor do olho, CPF, nome, endereço etc.

O objetivo do seu código é tratar o "ser humano" como cliente. Para cada cliente é preciso obter as informações de CPF, Nome e Endereço. As outras informações que pode obter na classe "ser humano", como cor da pele, cor do olho, altura, peso etc. não são importantes para esta situação.

Portanto, podemos abstrair estas informações e considerar somente o que importa.

Encapsulamento

Encapsulamento

- O encapsulamento é um dos pilares da engenharia de software, especialmente na **programação orientada a objetos (POO)**. Ele se baseia na ideia de **esconder os detalhes de implementação de um objeto**, expondo apenas uma interface pública para interação.

Encapsulamento

- Ao encapsular algo, você está colocando um objeto dentro de um recipiente, igual a um remédio de cápsula.
- Quando você encapsula um objeto, você está criando uma proteção e um padrão. Com isso, o propósito é de proteger o desenvolvedor do código e o código do desenvolvedor.

Encapsulamento

- Uma classe **encapsula** comportamentos.

Exemplo

```
public class DiscoVoador {  
    public String cor;  
    private void darPartida() {  
        // Implementacao  
    }  
    private void irParaFrente(int metros) {  
        // Implementacao  
    }  
}
```

Encapsulamento

- Uma classe **encapsula** comportamentos.

Exemplo

```
public class DiscoVoador {  
    public String cor;  
    private void darPartida() {  
        // Implementacao  
    }  
    private void irParaFrente(int metros) {  
        // Implementacao  
    }  
}
```

Atributo

Métodos

A classe "DiscoVoador" encapsula o comportamento de um disco voador. O usuário não precisa saber como o disco voador se move ou como ele é ligado. Ele só precisa saber que o disco voador pode se mover e ser ligado.

Na pratica

- Classe: O nome da classe é "DiscoVoador". Uma classe define um modelo para a criação de objetos.
- Atributos: A classe possui um atributo chamado "cor". Um atributo representa um dado ou característica de um objeto.
- Métodos: A classe possui dois métodos: "darPartida()" e "irParaFrente()". Um método representa uma ação ou comportamento que um objeto pode realizar..

Resumo

1. Anteriormente, você aprendeu sobre abstração, certo? Que nada mais é do que simplificar uma classe, se concentrando apenas em informações importantes e relevantes para o propósito do código.
2. No encapsulamento esse conceito é muito importante, já que os métodos da classe encapsulada serão abstratos, ou seja, os métodos abstratos são previstos ali, mas não são implementados na interface.

Na pratica

Para entender melhor esse processo, pense, por exemplo, nas máquinas de café expresso nas padarias.



Na pratica

Como ela é feita, não é algo que precisamos saber para poder obter o resultado do processo (nosso café quentinho).

Por isso, a máquina é um objeto encapsulado, onde os ingredientes e o mecanismo do preparo do café ficam escondidos atrás da interface externa da máquina, composta por botões de comando (como ligar, desligar, tipos de bebida etc.) que auxiliam na preparação.



Herança

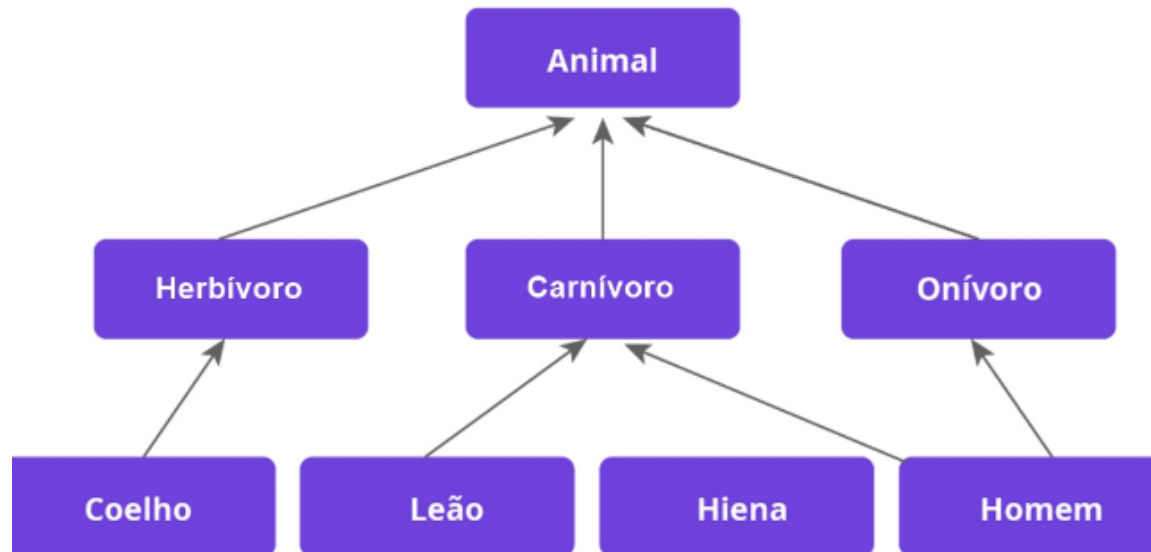
Herança

Assim como no mundo real, a herança na POO também diz respeito à **ação de herdar**. Ela nada mais é do que um objeto poder ser criado em uma outra classe, **levando consigo todos os atributos já existentes** em sua classe de origem.

A herança é uma maneira de reutilizar o código já existente em uma nova classe. Desta forma, o código é aprimorado com novas e melhores capacidades.

Herança

O organograma, mostra um exemplo de herança. Ao analisá-la, percebe-se que as classes herbívoro, carnívoro e onívoro podem herdar quaisquer atributos necessários da classe animal: tamanho, raça, cor etc. Da mesma forma, as classes leão, hiena, homem e coelho podem herdar atributos das classes herbívoro, carnívoro ou onívoro.



Polimorfismo

Polimorfismo

O polimorfismo na orientada a objetos permite que objetos de diferentes classes possam responder a uma mesma mensagem ou chamada de método de forma específica, de acordo com sua implementação particular.

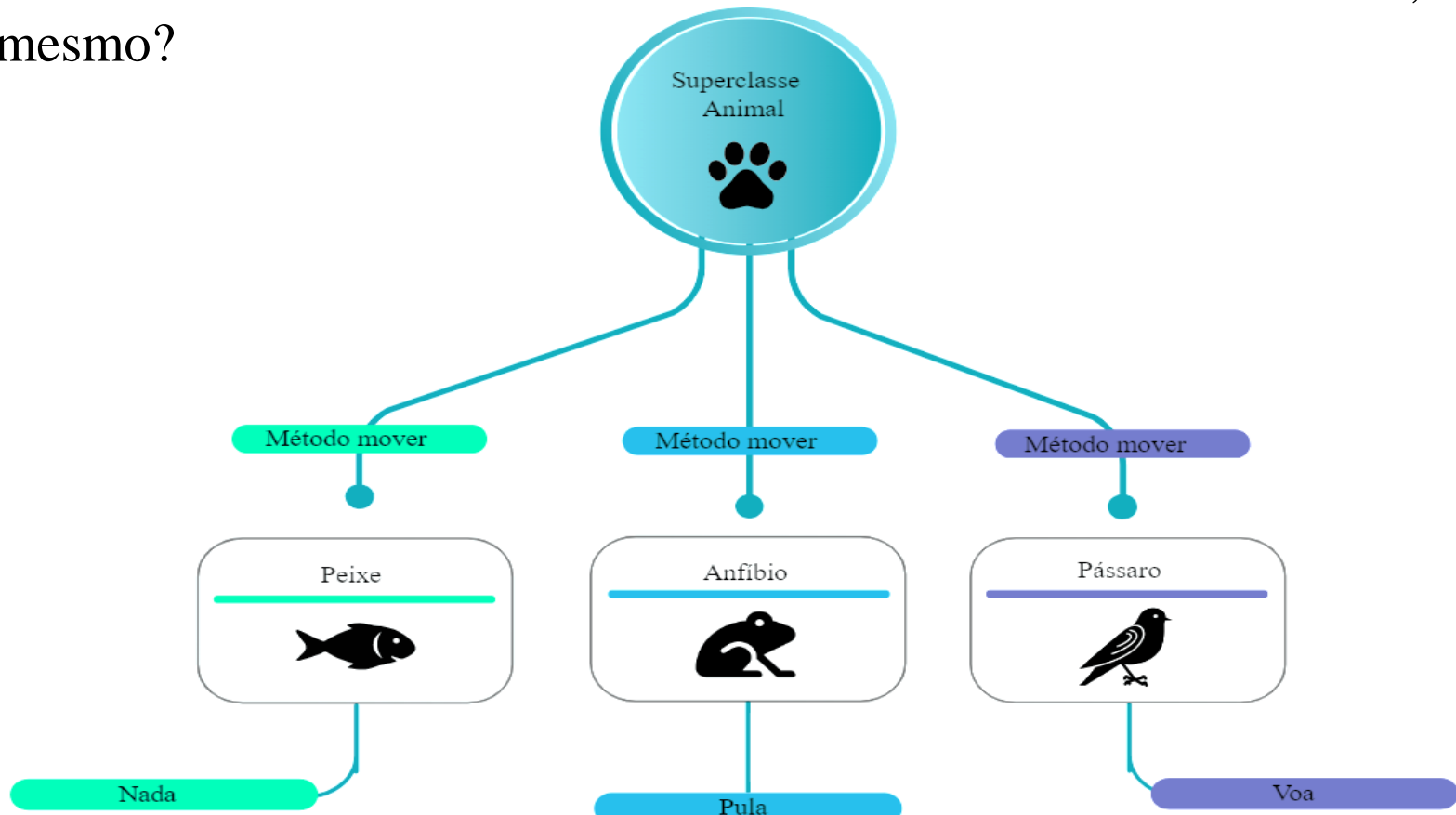
Isso significa que um mesmo método pode ter diferentes comportamentos em diferentes objetos, possibilitando tratar objetos de diferentes classes de forma uniforme.

Polimorfismo

É utilizado para que duas classes façam uso do mesmo método, implementando-o de formas diferentes. Literalmente, “polimorfismo” significa “muitas formas”. No contexto e projeto orientado a objetos, entretanto, refere-se à habilidade de uma variável de objeto de assumir formas diferentes.

Polimorfismo

Note que na superclasse Animal, as classes (peixe, anfíbio e pássaro) obedecem a um mesmo comando (método **mover**), mesmo que de maneiras diferentes? Assim fica mais fácil entender o conceito, não é mesmo?



Exercício Extra 1

Ano: 2009 Banca: FCC Órgão: SEFAZ-SP Prova: FCC - 2009 - SEFAZ-SP - Agente Fiscal de Rendas - Tecnologia da Informação - Prova 3

Uma classe é uma abstração que ajuda a lidar com a complexidade e um bom exemplo de abstração é:

- A - Um aluno e as disciplinas que está cursando.
- B - Um professor e os cursos nos quais ministra aulas.
- C - Um funcionário e o departamento em que trabalha.
- D - Uma pessoa e o número do seu CPF na Receita Federal.
- E - Uma casa e a empresa que a projetou e construiu.

Exercício Extra 1

Ano: 2009 Banca: FCC Órgão: SEFAZ-SP Prova: FCC - 2009 - SEFAZ-SP - Agente Fiscal de Rendas - Tecnologia da Informação - Prova 3

Uma classe é uma abstração que ajuda a lidar com a complexidade e um bom exemplo de abstração é:

- A - Um aluno e as disciplinas que está cursando.
- B - Um professor e os cursos nos quais ministra aulas.
- C - Um funcionário e o departamento em que trabalha.
- D - Uma pessoa e o número do seu CPF na Receita Federal.**
- E - Uma casa e a empresa que a projetou e construiu.

Explicação

Para entender a questão é necessário compreender o conceito de abstração em Engenharia de Software, especialmente na Orientação a Objetos. **Abstração é o processo de reduzir a complexidade de um sistema** ao definir classes que representam características e comportamentos relevantes de objetos do mundo real de maneira simplificada, **ignorando** detalhes menos importantes para o contexto em questão.

Explicação

A razão pela qual a alternativa D está correta, está na representatividade do CPF como uma abstração para a entidade pessoa. O CPF é uma forma de identificação única de uma pessoa perante o sistema da Receita Federal, independente de outras características como nome, ocupação, ou relações pessoais. **Ele simplifica a maneira como o governo lida com os cidadãos**, focando apenas no necessário para identificação e controle fiscal.

Exercício Extra 2

Ano: 2013 Banca: CESPE / CEBRASPE Órgão: TRE-MS Prova: CESPE - 2013 - TRE-MS - Técnico Judiciário - Programação de Sistemas

Assinale a opção correta quanto à abordagem conceitual de abstração sob o paradigma de programação orientada a objetos.

A - As abstrações, idealmente, caracterizam-se por não serem grandes demais em comparação aos módulos, pois senão elas se tornam multifuncionais e de difícil compreensão. Como consequência, a abstração deve ser implementada apenas no nível de estruturas de dados necessários para se atingir o objetivo pretendido.

B - Abstração consiste em uma linguagem puramente lógica. A motivação para isso veio em parte da vontade de se reconciliar o uso da lógica como uma linguagem declarativa de representação do conhecimento com a representação procedimental do conhecimento.

C - Abstração é uma linguagem declarativa que permite acesso à base de dados mediante a utilização da teoria dos conjuntos e da álgebra relacional como fundamento de seu funcionamento.

D - Abstração é um conceito segundo o qual o sistema ou software é dividido em partes distintas. Compõe o ferramental necessário para um programa mais legível com uma melhor manutenção e melhor desempenho por meio da programação orientada a objetos.

E - Abstração é a habilidade de se concentrar nos aspectos essenciais de um contexto qualquer, ignorando características menos importantes ou acidentais. Em modelagem orientada a objetos, uma classe é uma abstração de entidades existentes no domínio do sistema de software.

Exercício Extra 2

Ano: 2013 Banca: CESPE / CEBRASPE Órgão: TRE-MS Prova: CESPE - 2013 - TRE-MS - Técnico Judiciário - Programação de Sistemas

Assinale a opção correta quanto à abordagem conceitual de abstração sob o paradigma de programação orientada a objetos.

A - As abstrações, idealmente, caracterizam-se por não serem grandes demais em comparação aos módulos, pois senão elas se tornam multifuncionais e de difícil compreensão. Como consequência, a abstração deve ser implementada apenas no nível de estruturas de dados necessários para se atingir o objetivo pretendido.

B - Abstração consiste em uma linguagem puramente lógica. A motivação para isso veio em parte da vontade de se reconciliar o uso da lógica como uma linguagem declarativa de representação do conhecimento com a representação procedimental do conhecimento.

C - Abstração é uma linguagem declarativa que permite acesso à base de dados mediante a utilização da teoria dos conjuntos e da álgebra relacional como fundamento de seu funcionamento.

D - Abstração é um conceito segundo o qual o sistema ou software é dividido em partes distintas. Compõe o ferramental necessário para um programa mais legível com uma melhor manutenção e melhor desempenho por meio da programação orientada a objetos.

E - Abstração é a habilidade de se concentrar nos aspectos essenciais de um contexto qualquer, ignorando características menos importantes ou acidentais. Em modelagem orientada a objetos, uma classe é uma abstração de entidades existentes no domínio do sistema de software.

Explicação

A abstração é um princípio fundamental que permite ao desenvolvedor focar nas características relevantes de um objeto, desconsiderando os detalhes que não contribuem para o propósito específico do sistema em desenvolvimento. Em outras palavras, ela simplifica a complexidade do mundo real ao modelar classes que representam entidades com atributos e comportamentos essenciais para a lógica do programa.

Exercício Extra 3

Ano: 2009 Banca: UNIRIO Órgão: UNIRIO Prova: UNIRIO - 2009 - UNIRIO - Técnico de Tecnologia da Informação

O conceito de encapsulamento na orientação a objetos refere-se Alternativas

A - ao mecanismo pelo qual uma classe pode estender outra classe, aproveitando seus comportamentos (métodos) e variáveis possíveis (atributos).

B - à separação dos aspectos internos e externos de um objeto, restringindo o acesso direto ao estado do objeto.

C - à habilidade de concentrar nos aspectos essenciais de um contexto qualquer, ignorando características menos importantes ou acidentais.

D - ao princípio pelo qual classes distintas podem invocar métodos que têm a mesma assinatura, mas comportamentos especializados.

E - à utilização do mesmo nome para símbolos ou métodos com funcionalidades distintas, geralmente diferenciados pela sua assinatura.

Exercício Extra 3

Ano: 2009 Banca: UNIRIO Órgão: UNIRIO Prova: UNIRIO - 2009 - UNIRIO - Técnico de Tecnologia da Informação

O conceito de encapsulamento na orientação a objetos refere-se Alternativas

A - ao mecanismo pelo qual uma classe pode estender outra classe, aproveitando seus comportamentos (métodos) e variáveis possíveis (atributos).

B - à separação dos aspectos internos e externos de um objeto, restringindo o acesso direto ao estado do objeto.

C - à habilidade de concentrar nos aspectos essenciais de um contexto qualquer, ignorando características menos importantes ou acidentais.

D - ao princípio pelo qual classes distintas podem invocar métodos que têm a mesma assinatura, mas comportamentos especializados.

E - à utilização do mesmo nome para símbolos ou métodos com funcionalidades distintas, geralmente diferenciados pela sua assinatura.

Explicação

Ao falarmos de encapsulamento, estamos nos referindo diretamente a esse mecanismo de ocultação de estado. Um objeto encapsulado é como uma "caixa preta" que esconde seus dados internos e fornece operações que podem ser realizadas com ele, sem revelar como estas operações são realizadas internamente.

ATÉ A PRÓXIMA!

Bibliografia básica

- BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. UML: guia do usuário. Rio de Janeiro, RJ: Campus, 2005.
- DEITEL, H. M.; DEITEL, P.J. Java: como programar. 6. ed. Porto Alegre, RS: Bookman, 2005.
- GUEDES, G. T. A. UML 2: uma abordagem prática. São Paulo, SP: Novatec, 2009.

Bibliografia complementar

- BARNES, D. J.; KOLLING, M. Programação orientada a objetos com Java. 4. ed. São Paulo, SP: Pearson Prentice Hall, 2009.
- BRUEGGE, B.; DUTOIT, A. H. Object-oriented software engineering: using UML, patterns, and Java. 2. ed. Upper Saddle River, NJ: Prentice Hall, 2003.
- FOWLER, M. UML essencial: um breve guia para a linguagem-padrão de modelagem de objetos. 3. ed. Porto Alegre, RS: Bookman, 2005.
- LARMAN, C. Utilizando UML e padrões: uma introdução à análise e ao projeto orientados a objetos e ao desenvolvimento iterativo. 3. ed. Porto Alegre, RS: Bookman, 2007.